2009 International Conference on Network-Based Information Systems

Locals Search Algorithms for Efficient Router Nodes Placement in Wireless Mesh Networks

Fatos Xhafa
Department of Languages and Informatics Systems
Technical University of Catalonia
Campus Nord, C/Jordi Girona 1-3
08034 Barcelona, Spain
EMail: fatos@lsi.upc.edu

Christian Sánchez
Department of Languages and Informatics Systems
Technical University of Catalonia
Campus Nord, C/Jordi Girona 1-3
08034 Barcelona, Spain
EMail: csanchez@lsi.upc.edu

Leonard Barolli
Department of Information and Communication Engineering
Fukuoka Institute of Technology
3-30-1 Wajiro-higashi, Higashi-ku
Fukuoka 811-0295, Japan
EMail: barolli@fit.ac.jp

Abstract

Wireless Mesh Networks (WMNs) are an important networking infrastructure for providing cost-efficient broadband wireless connectivity to a group of users. WMNs are increasingly being used in urban areas, metropolitan and municipal area networks for deployment of medical, transport, surveillance systems, etc. The good performance and operability of WMNs largely depends on placement of mesh routers nodes in the geographical area to achieve network connectivity and stability. Therefore, optimal placement of router nodes in WMNs is a main issue in deployment of WMNs. In this work we propose and evaluate local search methods for placement of mesh routers in WMNs with a two fold objective: maximizing the size of the giant component in the network and user coverage. Given a grid area where to distribute a given number of mesh router nodes, which can have different radio coverage, and a number of fixed clients a priori distributed in the given area, local search methods explore different local movements and incrementally improve the quality of the router nodes placement in terms of network connectivity and user coverage. We have experimentally evaluated the considered local search methods through a benchmark of generated instances varying from small to large size. In the instances, different distributions of mesh clients (Uniform, Normal, Exponential and Weibull) are considered aiming to evaluate the quality of achieved solutions for different client distributions. Although local search methods can be stuck in local optima, the experimental evaluation showed their good performance; local search methods could thus be applied as a first choice for optimizing network connectivity and user coverage in WMNs.

1. Introduction

With the fast development of wireless technologies, Wireless Mesh Networks (WMNs) [1] are becoming an important networking infrastructure due to their low cost and increased high speed wireless Internet connectivity. In a WMN we have two types of nodes: mesh routers and mesh clients. Mesh routers are similar to normal routers but incorporate also additional functions to support mesh networking, and are usually equipped with multiple interfaces to work with different wireless technologies. Another feature of this type of routers with respect to usual routers is their ability to provide the same coverage with much less transmitter power through multi-hop communications. Also, mesh routers can be installed on a dedicated machine or on a general purpose machine. With regard to mesh clients, they have the necessary functions for mesh networking and could also be able to act as routers but do not have the functionality of a gateway or bridge and their single wireless interface with the hardware and software platform is much simpler than in the case of mesh routers.

Fast development of WMNs is pushed by their low cost nature that makes them an economical alternative for providing wireless Internet connectivity, especially in developing countries, avoiding costs of deployment and maintenance of wired Internet infrastructures. Applications of WMNs include WMNs for urban areas, community networking, metropolitan area networks, municipal wireless mesh networks, corporative networks, medical systems, transport systems, surveillance systems, etc. [2]. In all these applications, WMNs provide cost-efficient broadband wireless Internet connectivity to a group of users. As a case study of the use

978-0-7695-3767-2/09 $25.00 © 2009 IEEE
DOI 10.1109/NBiS.2009.21

of such networks we can highlight the "*One hundred dollar laptop*" project developed at MIT for schools in developing countries. The objective of the project is to establish a mesh network to create a robust and inexpensive infrastructure from students' laptops. The connections made by the laptops would not need an internal infrastructure, like Internet, since one of the computers on the grid may share the connection with the neighboring nodes. Another project in this spirit is the one promoted by the UN "*One Laptop per Child*" that would permit to implement this type of networking.

Mesh network topology distinguishes for providing reliability, robustness, and self-configuring properties achieved through multiple redundant communications paths in the network. The placement of mesh nodes plays an important role in achieving such properties. Indeed, the performance of WMNs is primarily affected by the location of mesh nodes, specifically, that of mesh router nodes of the WMN. However, in a real deployment of WMN the automatic or purely random node placements produce poor performance WMN since the resulting placement could be far from optimal. Further, real deployment of WMNs may require to take into account specific restrictions and characteristics of real geographic area and therefore one needs to explore different topologies for placing mesh routers. In fact, node placement can be seen as a crucial design and management issue in WMNs.

Mesh node placement in WMN can be seen as a family of problems. Different versions of the problem can be obtained depending on the types of mesh nodes to deploy as well as the objectives to optimize. For instance, in [6], [11], [8] is considered the gateway placement aiming to optimize the throughput. In [3], the authors consider a bi-objective version of the problem for two-tier WMNs.

Node placement belongs to the family of placement problems, which are shown (through graph theoretic approaches or placement problems, e.g. [4], [5]) computationally hard to solve for most of the formulations [12]. In fact, the node placement problem considered here is even more challenging due to two additional characteristics: (a) locations of mesh router nodes are not pre-determined (any available position in the considered area can be used for deploying the mesh routers), and (b) routers are assumed to have their own radio coverage area.

In this work we consider the version of the problem that given an area where to distribute a number of mesh router nodes and a number of mesh client nodes of fixed positions (of an arbitrary distribution), finds a location assignment for the mesh routers that maximizes the network connectivity (size of the giant component) and client coverage. These two objectives are among most important objectives in WMNs. Both of them are related to the performance of the network; the later can be also seen as a QoS in WMNs.

We consider approaching the mesh router nodes placement using local search methods as optimization methods in which the network connectivity and user coverage are measured and evaluated. The optimization approach follows a hierarchical setting in which the primary objective is that of maximizing the size of the giant component and user coverage is considered secondary one. In such setting, the local search method tries to first maximize the size of the giant component and then tries to maximize the user coverage without worsening the size of the giant component.

Although in general local search methods can get stuck in local optima, they are effective optimization methods and thus can be considered a first choice for computing near-optimal placement of mesh router nodes in WMNs. We have experimentally evaluated the proposed local search methods through a benchmark of generated instances, consisting of 48 instances, ranging from small to large size in terms of mesh router nodes and the grid area. Moreover, instances are generating using different distributions of mesh clients (Uniform, Normal, Exponential and Weibull). In the experimental study we evaluated the effectiveness of different local movements, namely, *Random*, *Radius*, *Swap* and *Combination*, in terms of the maximization of the size of the giant component and user coverage.

The rest of the paper is organized as follows. In Section 2, we briefly mention the main architectures for WMNs. The problem under study is defined in Section 3. The considered local search method, namely the Hill Climbing algorithm, is presented in Section 4 and its application to mesh router nodes placement problem in Section 5. The experimental evaluation is given in Section 6. We end the paper in Section 7 with some conclusions.

2. Architectures of WMNs

The architecture of the nodes in WMNs can be classified according to the functionalities they offer as follows:

Infrastructure/Backbone WMNs: This type of architecture (also known as infrastructure meshing) is the most used and consists of a grid of mesh routers which are connected to different clients. Moreover, routers have gateway functionality thus allowing Internet access for clients. This architecture enables integration with other existing wireless networks and is widely used in neighboring communities. We will consider this kind of architecture in this work.

Client WMNs: Client meshing architecture provides a communications network based on peer-to-peer over client devices (there is no the role of mesh router). In this case we have a network of mesh nodes which provide routing functionality and configuration as well as end-user applications, so that when a packet is sent from one node to another, the packet will jump from node to node in the mesh of nodes to reach the destination.

Hybrid WMNs: This architecture combines the two previous ones, so that mesh clients are able to access the network through mesh routers as well as through direct

connection with other mesh clients. Benefiting from the advantages of the two architectures, Hybrid WMNs can connect to other networks (Internet, Wi-Fi, and sensor networks) and enhance the connectivity and coverage due to the fact that mesh clients can act as mesh routers.

3. Problem Definition

We consider the version of the mesh node placement problem in which, given an area where to distribute a number of mesh router nodes and a number of mesh client nodes of fixed positions (of an arbitrary distribution), finds a location assignment for the mesh routers that maximizes the network connectivity (size of the giant component) and client coverage. An instance of the problem consists of:

- N mesh router nodes, each having its own radio coverage, defining thus a vector of routers.
- An area $W \times H$ where to distribute N mesh routers. Positions of mesh routers are not pre-determined. The area is divided in square cells of an *a priori* fixed length and mesh router nodes are to be deployed in the cells of the grid area.
- M client mesh nodes located in arbitrary cells of the considered grid area, defining a matrix of clients.

An instance of the problem can be formalized by an adjacency matrix of the WMN graph, whose nodes are of two types: router nodes and client nodes and whose edges are links in the mesh network.

The objective is to place mesh router nodes in cells of considered area to maximize network connectivity and user coverage. In this work, the network connectivity is measured through the size of the giant component in the WMN. Network connectivity and user coverage are among most important metrics in WMNs. The former measures the degree of connectivity of the mesh nodes while the later refers to the number of mesh client nodes connected to the WMN. Both objectives are important and directly affect the network performance; nonetheless, network connectivity is considered as more important than user coverage. It should also be noted that in general optimizing one objective could effect the other objective although there is no direct relation among these objectives nor are they contradicting.

Optimization setting. For optimization problems having two or more objective functions, two settings are usually considered: the hierarchical and simultaneous optimization. In the former, the objectives are classified (sorted) according to their priority. Thus, for the two objective case, one of the objectives, say f_1, is considered as primary objective and the other, say f_2, as secondary one. The meaning is that the optimization is carried out in two steps: in the first we try to optimize f_1, and then, we try to optimize f_2 without worsening the best value of $f1$. In the later approach, both objectives are optimized simultaneously.

In this work we have considered the hierarchical approach in which the size of the giant component is a primary objective and the user coverage is a secondary one. Thus, the local search algorithm will first maximize the size of the giant component through local perturbations; next, when no further improvements are possible, the algorithm will try to maximize the user coverage without worsening the size of the giant component. In such approach, the network connectivity (through the maximization of size of giant component) is considered as most important since connectivity of the network is crucial for WMNs.

It should be noticed from the above problem description that mesh client nodes can be arbitrarily situated in the given area. For evaluation purposes, it is interesting, however, to consider concrete distributions of clients. For instance, it has been shown from studies in real urban areas or university campuses that users (client mesh nodes) tend to cluster to hotspots. Therefore different client mesh nodes distributions should be considered, for instance Weibull distribution, in evaluating WMN metrics.

We have considered Uniform, Normal, Exponential and Weibull distributions for client mesh nodes in the experimental evaluation (see Section 6).

4. Hill Climbing Algorithm

Hill Climbing (HC) is local search algorithm and is based on incremental improvements of solutions as follows: it starts with a solution (which may be randomly generated or *ad hoc* computed) considered as the current solution in the search space. The algorithm examines its neighboring solutions and if a neighbor is better than current solution then it can become the current solution; the algorithm keeps moving from one solution to another one in the search space until no further improvements are possible. There are several variants of the algorithm depending on whether a simple climbing, steepest ascent climbing or stochastic climbing is done:

- *Simple climbing*: the next neighbor solution is the first that improves current solution.
- *Steepest ascent climbing*: all neighbor solutions are examined and the best one is chosen as next solution.
- *Stochastic climbing*: a neighbor is selected at random, and according to yielded improvement of that neighbor is decided whether to choose it as next solution or to examine another neighbor. This kind of climbing has more general forms known as Metropolis and Simulated Annealing algorithms.

Of course, for each climbing method described above, several versions of the algorithm are possible depending on the way a neighbor solution is selected, that is, depending on the neighborhood structure used.

It should be noted that Hill Climbing usually ends up in local optima, which can be overcome in some cases by

adopting additional techniques such as: (a) getting back to a previous state and exploring another direction; (b) jumping to a new solution, possibly "far away" from current solution; and, (c) considering several search direction in solution space at the same time.

Despite its limitations, HC is an interesting choice to cope in practice with hard combinatorial optimization problems; it is simple to implement and works well for practical purposes.

We present the pseudo-code of HC in Algorithm 1. The algorithm first generates a solution which serves as starting point in the search space. Then, the algorithm iteratively selects a movement based on current solution, evaluates the resulting movement in terms of possible improvements with respect to current solution. If the resulting neighbor improves fitness of current solution, the current solution is moved to the new neighbor and so on. In the algorithm the function $\delta(s, m)$ computes the improvement yielded by applying movement m to current solution s (as usually, for maximization, a positive value of δ function means improvement with respect to fitness of current solution).

Algorithm 1 Hill Climbing algorithm for maximization. f is the fitness function.

1: **START**: Generate an initial solution s_0;
2: $s = s_0$; $s^* = s_0$; $f^* = f(s_0)$;
3: **repeat**
4: **MOVEMENT SELECTION**: Choose a movement $m = select\_movement(s)$;
5: **EVALUATE & APPLY MOVEMENT**:
6: **if** $\delta(s, m) \geq 0$ **then**
7: $s' = appply(m, s)$;
8: $s = s'$;
9: **end if**
10: **UPDATE BEST SOLUTION**:
11: **if** $f(s') > f(s^*)$ **then**
12: $f^* = f(s')$;
13: $s^* = s'$;
14: **end if**
15: **return** s^*, f^*;
16: **until** (stopping condition is met)

5. Application of Hill Climbing to Mesh Routers Node Placement

As can be seen from Algorithm 1, Hill Climbing is a generic search method and can be instantiated for any combinatorial optimization problem. To this end, one needs to specify the entities of the algorithm with specific information of the problem at hand. We present next the generation of the initial solution, the computation model of the fitness function as well as the definitions of local movements for the Mesh Routers Node Placement problem.

5.1. Initial solution

The first step of the algorithm is to generate the initial solution that will serve as a starting point of the search trajectory in the solution space. In most implementations of HC in literature the starting point is randomly generated. As random generation could in general yield poor solutions, more specific methods can be used. In our case, we can use any of *ad hoc* methods presented in Xhafa et al. [10].

5.2. Fitness function

An important aspect in implementing HC algorithm is the definition of the objective function which will guide the search towards promising areas of solution space as well as the appropriate encoding. Ideally we would be interested to build functions with some "regularity" so that we can easily verify that for two solutions that are close in the search space, their respective values in the objective function are similar. One issue to consider is that if the fitness function has not been coded correctly, many local optima can appear in the search space preventing the search progress towards (near-)optimal solutions.

In the case of Mesh Routers Node Placement problem, we are dealing with an optimization problem with multiple criteria – maximizing size of giant component and user coverage. These two objectives are optimized using a hierarchical approach in which the size of the giant component is a primary objective and the user coverage is a secondary one. In such approach, we first try to optimize the size of the giant components and then, we try to improve user coverage without worsening the value of the size of the giant component. As can be seen, in our approach the size of the giant component is considered of having more priority in comparison to user coverage. The rationale behind this approach is that the size of the giant component directly relates to the connectivity and stability of the wireless network.

5.3. Local movement types

The definition of the local movement is crucial to any local search algorithm. The local movement is in charge to perform a small perturbation to the current solution yielding thus a new solution, which is called neighbor solution. Thus, the definition of local movement implies a neighborhood structure for the problem at hand. Local movements are usually defined based on combinatorial properties of the problem at hand, the most common one being flipping the value of a position, swapping values of two positions, etc. in the combinatorial structure of the solution.

In the implementation of HC, we defined three different types of movements, namely, *Random*, *Radius* and *Swap*.

We also considered a fourth movement type which is a combination of Random, Radius and Swap movements.

- *Random*: this movement chooses a router at random and places it to a new position selected at random in the grid area.
- *Radius*: this movement selects the router of largest radio coverage and places it in the most dense area in terms of number of client mesh nodes of the grid area (see Algorithm 2). This movement could yield better performance but has the drawback of concentrating mesh routers in the most dense area of clients.
- *Swap*: this movement consists in exchanging the placement of two routers. More precisely, the worst router (that of smallest radio coverage) in the most dense area (in terms of number of client mesh nodes) is exchanged with the best router (that of largest radio coverage) of the sparsest area (see Algorithm 3). The idea is to promote the placement of best routers in most dense areas of the grid area.
- *Combination*: in this movement we consider a composition of previous movements in blocks yielding to a larger sequence: $< Rand_1, \ldots, Rand_k; Radius_1, \ldots, Radius_k; Swap_1, \ldots, Swap_k >$, where k is a user specified parameter. This movement would in particular be useful to speed up the search towards local optima.

Algorithm 2 Radius movement.

1: Choose values H_g and W_g for height and width of a small grid area.
2: Compute the position of most dense $H_g \times W_g$ area and (x_{dense}, y_{dense}) its central point.
3: Compute the position of the router of largest radio coverage $(x_{largest\_coverage}, y_{largest\_coverage})$.
4: Move router at $(x_{largest\_coverage}, y_{largest\_coverage})$ to new position (x_{dense}, y_{dense}).
5: Re-establish mesh nodes network connections.

Algorithm 3 Swap movement.

1: Choose values H_g and W_g for height and width of a small grid area.
2: Compute the position of most dense $H_g \times W_g$ area.
3: Compute the position (x_{dense}, y_{dense}) of less powerful router within the dense area.
4: Compute the position of most sparse $H_g \times W_g$ area.
5: Compute the position (x_{sparse}, y_{sparse}) of most powerful router within the sparse area.
6: Swap routers in (x_{dense}, y_{dense}) and (x_{sparse}, y_{sparse}) positions.
7: Re-establish mesh nodes network connections.

5.4. Acceptability of neighboring solutions

As explained earlier, the acceptability of neighboring solutions can done in different ways (simple ascent, steepest ascent, or stochastic). In our case, we have adopted the simple ascent, that is, if s is current solution and m is a movement, the resulting solution s' obtained by applying m to s will be accepted, and hence become current solution, iff the fitness of s' is at least as good as fitness of solution s. In terms of δ function, s' is accepted and becomes current solution if $\delta(s, m) \geq 0$. It should be noted that in this definition we are also accepting solutions that has the same fitness as previous solutions. The aim is to give chances to the search to move towards better solutions in solution space. A more strict version would be to accept only solutions that improve the fitness function ($\delta(s, m) > 0$).

5.5. Implementation of Hill Climbing

The implementation of the Hill Climbing is done using a skeleton approach. In this approach, a generic implementation of the main flow of the HC methods is provided. Other entities, such as Problem, Movement and Solution are interfaces further instantiated for the Mesh Routers Nodes problem. In particular Movement entity allows to specify different types of local movements that can be easily plugged in the HC method. We present in Fig. 1 the UML diagram of the skeleton for HC method.

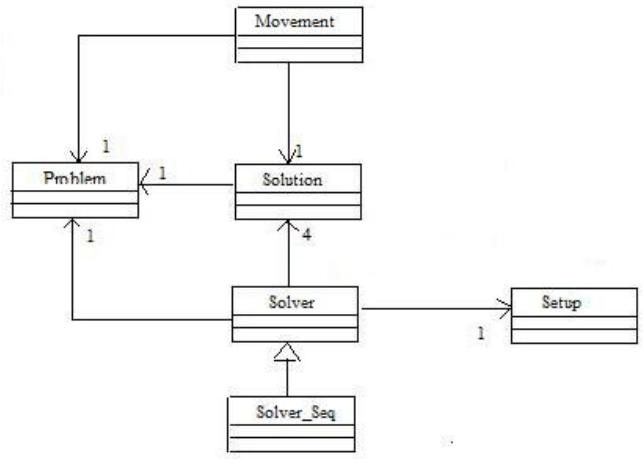

Figure 1. Skeleton of the Hill Climbing method.

6. Experimental Study

In this section we present computational results for the performance of the local search algorithms. Initially we used randomly generated instances of the problem to tune the parameters and study the performance of each local movement. Then, we used a benchmark of instances to evaluate

the performance of the local search method measured in terms of the size of the giant component and user coverage objectives.

6.1. Benchmark of instances

We have generated a benchmark consisting of 48 instances, having different sizes of grid area and using four probability distributions for the positions of mesh client nodes in the grid area. Instances are arranged in three groups, each having 16 instances and are labelled $I_{x\times x\_D\_k}$, where:

- x stands for the height and width of the grid area, that is, the number of cells of arbitrary edge length; it takes values 32, 64 and 128.
- D stands for the distribution of the client mesh routers in the grid area; four distributions are considered: Uniform (U), Normal (N), Exponential (E) and Weibull (W).
- k is the index of the instance.

Thus, we have 16 instances for each grid size (32, 64 and 128, resp.) and within each group we have 4 instances for each distribution (Uniform, Normal, Exponential and Weibull, resp). For instance, in this notation, $I_{64\times 64\_N\_3}$ denotes the third instance of a 64×64 grid area, with mesh clients nodes positions generated using Normal distribution.

Instances of 32×32 grid area consist of 16 mesh routers nodes and 48 client mesh nodes; instances of 64×64 grid area consist of 32 mesh routers nodes and 96 client mesh nodes; and, instances of 128×128 grid area consist of 64 mesh routers nodes and 192 client mesh nodes.

6.2. Study of the performance of local movements

A preliminary experimental study was conducted to evaluate the performance of the four local movements, namely, Random, Radius, Swap and Combination. For the purposes of this preliminary study we used randomly generated instances of different sizes, namely grid areas of 32×32, 64×64, 128×128 sizes respectively. It should be noted that to avoid biased results in the experimental study of the Hill Climbing, these instances are not instances of the benchmark. Moreover, because the algorithm takes random decisions, 15 independent runs have been carried out for the same instance and the performance is evaluated.

As a result of this preliminary study we concluded that:

- for instances of 32×32 grid area, the *Radius* is the local movement that quickly reaches better values of the size of giant component. However, *Combination* is the first to establish a network in which all routers are connected. Thus, for small size instances, Combination seems an appropriate local movement in Hill Climbing algorithm.
- for instances of 64×64 grid area, *Swap* shows the best performance. *Combination* in this case does not work as well as in the case of 32×32 grid.
- for instances of 128×128 grid area, again, *Swap* shows the best performance, followed by Radius. Combination in this case performs worst.

6.3. Computational results

We report computational results for the Hill Climbing algorithm for benchmark instances. Again, 15 independent runs have been carried out for the same instance and results (mean value and standard deviation) are reported for both the size of giant component and user coverage.

Results for 32×32 size instances. We present in Table 1 computational results for benchmark instances of 32×32 grid size (*Combination* is used as local movement).

Table 1. Size of giant component and user coverage for 32×32 grid size instances, 16 routers and 48 clients.

Instance	Size of giant component			Users covered		
	best	avg	dev	best	avg	dev
I32x32_U_1	11	8	0.3	24	23	0.1
I32x32_U_2	14	9	0.5	23	24	0.1
I32x32_U_3	15	9	0.6	15	22	0.7
I32x32_U_4	12	8	0.4	20	24	0.4
I32x32_N_1	16	15	0.1	43	39	0.4
I32x32_N_2	16	15	0.1	42	41	0.1
I32x32_N_3	16	15	0.1	42	40	0.2
I32x32_N_4	16	15	0.1	43	39	0.4
I32x32_E_1	16	14	0.2	42	39	0.3
I32x32_E_2	16	12	0.4	46	39	0.7
I32x32_E_3	16	12	0.4	44	37	0.7
I32x32_E_4	16	13	0.3	44	35	0.9
I32x32_W_1	16	12	0.4	41	31	1
I32x32_W_2	16	13	0.3	45	35	1
I32x32_W_3	16	13	0.3	45	33	1.2
I32x32_W_4	16	12	0.4	40	33	0.7

As can be seen from the table, routers connectivity is achieved for almost all instances of the benchmark. However, *Combination* movement works best for Exponential and Weibull distribution of client mesh nodes. We show in Fig. 2, the graphical representation of the performance of the four movements for the $I_{32\times 32\_U\_3}$ instance for the size of giant component.

Results for 64×64 size instances. We present in Table 2 computational results for benchmark instances of 64×64 grid size (*Swap* is used as local movement).

As can be seen from the table, routers connectivity is achieved for almost all instances of the benchmark. However, *Swap* movement works best for Exponential and Weibull distribution of client mesh nodes. For Uniform distribution of client nodes, *Swap* movement doesn't perform well. We show in Fig. 3, the graphical representation of the performance of the four movements for the $I_{64\times 64\_U\_3}$ instance for the size of giant component.

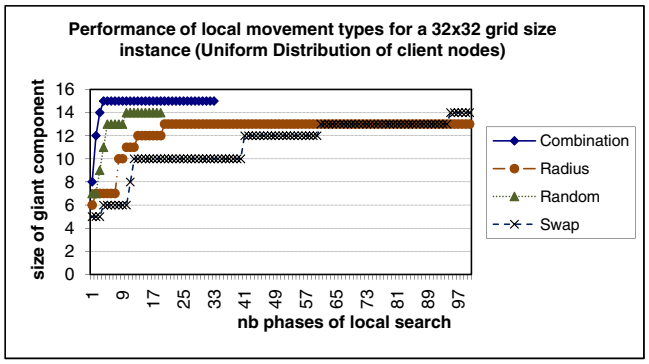

Figure 2. The size of giant component for the $I_{32\times32\_U\_3}$ instance computed by Hill Climbing using four local movements.

Table 2. Size of giant component and user coverage for 64×64 grid size instances, 32 routers and 96 clients.

Instance	Size of giant component			Users covered		
	best	avg	dev	best	avg	Dev
I64x64_U_1	9	9	0	59	59	0
I64x64_U_2	15	10	0.5	45	46	0.1
I64x64_U_3	17	17	0	55	55	0
I64x64_U_4	15	15	0	51	51	0
I64x64_N_1	32	32	0	81	81	0
I64x64_N_2	32	32	0	90	90	0
I64x64_N_3	32	32	0	90	90	0
I64x64_N_4	32	32	0	85	85	0
I64x64_E_1	32	32	0	89	89	0
I64x64_E_2	32	32	0	92	92	0
I64x64_E_3	31	31	0	93	93	0
I64x64_E_4	32	32	0	87	87	0
I64x64_W_1	32	27	0.5	93	87	0.6
I64x64_W_2	32	29	0.3	91	88	0.3
I64x64_W_3	32	24	0.8	92	87	0.5
I64x64_W_4	32	31	0.1	95	92	0.3

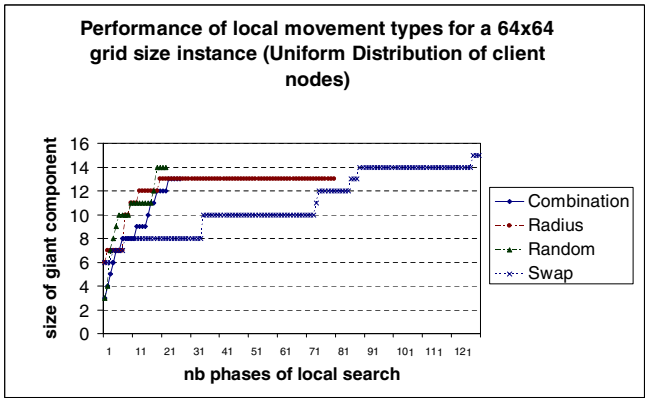

Figure 3. The size of giant component for the $I_{64\times64\_U\_3}$ instance computed by Hill Climbing using four local movements.

128×128 size instances. We present in Table 3 computational results for benchmark instances of 128×128 grid size (*Swap* is used as local movement).

Table 3. Size of giant component and user coverage for 128×128 grid size instances, 64 routers nodes and 192 clients.

Instance	Size of giant component			Users covered		
	best	avg	dev	best	avg	Dev
I64x64_U_1	16	11	0.5	83	89	0.6
I64x64_U_2	17	10	0.7	83	90	0.7
I64x64_U_3	15	9	0.6	82	90	0.8
I64x64_U_4	17	13	0.4	78	87	0.8
I64x64_N_1	52	45	1.2	126	133	0.7
I64x64_N_2	53	40	1.3	120	126	0.6
I64x64_N_3	45	37	0.8	131	125	0.6
I64x64_N_4	51	36	1.5	115	123	0.8
I64x64_E_1	32	29	1.1	130	137	0.7
I64x64_E_2	47	37	1	169	160	0.9
I64x64_E_3	32	27	0.9	92	111	1.9
I64x64_E_4	38	28	1.4	159	162	0.3
I64x64_W_1	53	36	1.4	156	157	0.1
I64x64_W_2	52	41	1.2	160	156	0.4
I64x64_W_3	53	36	1.4	155	171	0.6
I64x64_W_4	51	40	1.3	146	154	0.8

In contrast to 32×32 and 64×64 grid size instances, *Swap* movement achieved best results and without falling in slow convergence. This could be explained by the large size of the neighborhood of 128×128 instances for which there are greater chances of finding a local maximum. For the Weibull distribution all instances converged to network connectivity.

We show in Fig. 4, the graphical representation of the performance of the four movements for the $I_{128\times128\_N\_4}$ instance for the size of giant component.

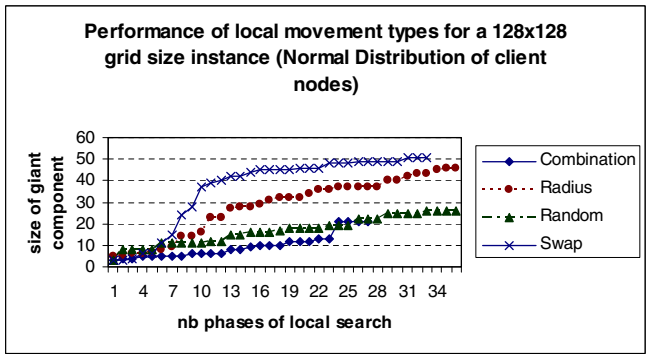

Figure 4. The size of giant component for the $I_{128\times128\_N\_4}$ instance computed by Hill Climbing using four local movements.

6.4. Summative evaluation

The experimental study revealed that Radius local movement performs very well for instances of small and medium

size grid area while for larger size instances, Swap movement is more efficient. This is so because Swap is too powerful (and computationally more costly) to carry it out in a grid of small size because Swap movement selects the router of smaller radius in the most densely populated region of clients and the router of the largest radio in less densely populated region of clients. Thus, in order for Swap to perform well, there should be enough diversity of mesh client nodes positions on the grid area. We also found that Combination showed a very good performance for instances of small size.

It should be noted however that the performance of the four local movements depends on the distribution of mesh client nodes. We have thus studied their performance for each of four distributions (Uniform, Normal, Exponential and Weibull). We observed that for Uniform distribution even a Random local movement performs well; however, for the rest of distributions Random performs poorly and Radius and Swap performed much better.

7. Conclusions

In this work we have presented local search based methods for the problem of optimal placement of mesh router nodes in Wireless Mesh Networks (WMNs). We consider the version of the problem in which a number of client mesh nodes are *a priori* distributed in a grid area, divided in small cells, in which mesh router nodes are to be deployed. The objective is to optimize the network connectivity and user coverage. Routers are assumed having their own radio coverage area.

In our approach, the objective is two fold: maximizing the size of the giant component in the network and user coverage in a hierarchical setting in which the size of the giant component is considered as a primary objective and the user coverage a secondary one. The proposed local search methods explore different local movements and incrementally improve the quality of the mesh router nodes placement in terms of network connectivity and user coverage. We have experimentally evaluated the considered local search methods through a benchmark of generated instances varying from small to large size. In the instances, different distributions of mesh clients (Uniform, Normal, Exponential and Weibull) are considered and the quality of achieved solutions for different client distributions are evaluated and reported. Although local search methods can be stuck in local optima, the experimental evaluation showed their good performance; moreover, we are able to identify the performance of the considered local movement for different types of mesh client nodes distributions. Thus, local search methods could be applied as a first choice for optimizing network connectivity and user coverage while deploying mesh router nodes in WMNs.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. Computer Networks **47**(4) (2005), 445-487.

[2] Ch. Chen and Ch. Chekuri. Urban Wireless Mesh Network Planning: The Case of Directional Antennas. Tech Report No. UIUCDCS-R-2007-2874, Department of Computer Science, University of Illinois at Urbana-Champaign, June 2007.

[3] A. Antony Franklin and C. Siva Ram Murthy. Node Placement Algorithm for Deployment of Two-Tier Wireless Mesh Networks. In Proceedings of IEEE GLOBECOM 2007, IEEE Global Communications Conference, November 2007, Washington, USA, pp. 4823-4827.

[4] M.R. Garey and D.S. Johnson. Computers and Intractability –A Guide to the Theory of NP-Completeness. Freeman, San Francisco, 1979.

[5] A. Lim, B. Rodrigues, F. Wang and Zh. Xua. $k-$Center problems with minimum coverage. Theoretical Computer Science **332** (2005) 1-17.

[6] S. N. Muthaiah and C. Rosenberg. Single Gateway Placement in Wireless Mesh Networks. In Proceedings of 8th International IEEE Symposium on Computer Networks, Turkey, June 2008.

[7] N. Nandiraju, D. Nandiraju, L. Santhanama, B. He, J. Wang, and D. Agrawal. Wireless mesh networks: Current challenges and future direction of web-in-the-sky. IEEE Wireless Communications, 79-89 (2007).

[8] M. Tang. Gateways Placement in Backbone Wireless Mesh Networks. International Journal of Communications, Network and System Sciences, **1**, 1-89. Published Online February 2009.

[9] T. Vanhatupa, M. Hännikäinen and T.D. Hämäläinen. Genetic Algorithm to Optimize Node Placement and Configuration for WLAN Planning. In Proceedings of 4th International Symposium on Wireless Communication Systems (ISWCS 2007), 612-616, 2007.

[10] Fatos Xhafa, Christian Sanchez, Leonard Barolli. Ad Hoc and Neighborhood Search Methods for Placement of Mesh Routers in Wireless Mesh Networks. Accepted at *The 11-th International Workshop on Multimedia Network Systems and Applications (MNSA-2009)*, held in conjunction with the IEEE 29-th International Conference on Distributed Computing Systems (ICDCS-2009), in June 22-26, 2009, Montreal, Quebec, Canada.

[11] P. Zhou, B. S. Manoj, and R. Rao. A gateway placement algorithm in wireless mesh networks. In Proceedings of the 3rd international Conference on Wireless Internet. Austin, Texas (2007).

[12] J. Wang, B. Xie, K. Cai and D.P. Agrawal. Efficient Mesh Router Placement in Wireless Mesh Networks, MASS 2007, Pisa, Italy (2007).